计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.20210551
59(3):518-532,2022

基于近端策略优化的阻变存储硬件加速器自动量化

魏 正[1] 张兴军[1] 卓志敏[2] 纪泽宇[1] 李泳昊[1]
[1](西安交通大学计算机科学与技术学院 西安 710049)
[2](北京电子工程总体研究所 北京 100854)
(frank.wei@stu.xjtu.edu.cn)

PPO-Based Automated Quantization for ReRAM-Based Hardware Accelerator

Wei Zheng[1], Zhang Xingjun[1], Zhuo Zhimin[2], Ji Zeyu[1], and Li Yonghao[1]
[1](*School of Computer Science and Technology, Xi'an Jiaotong University, Xi'an* 710049)
[2](*Beijing Institute of Electronic System Engineering, Beijing* 100854)

Abstract Convolutional neural networks have already exceeded the capabilities of humans in many fields. However, as the memory consumption and computational complexity of CNNs continue to increase, the "memory wall" problem, which constrains the data exchange between the processing unit and memory unit, impedes their deployments in resource-constrained environments, such as edge computing and the Internet of Things. ReRAM(resistive RAM)-based hardware accelerator has been widely applied in accelerating the computing of matrix-vector multiplication due to its advantages in terms of high density and low power, but are not adept for 32 b floating-point data computation, raising the demand for quantization to reduce the data precision. Manually determining the bitwidth for each layer is time-consuming, therefore, recent studies leverage DDPG(deep deterministic policy gradient) to perform automated quantization on FPGA(field programmable gate array) platform, but it needs to convert continuous actions into discrete actions and the resource constraints are achieved by manually decreasing the bitwidth of each layer. This paper proposes a PPO (proximal policy optimization)-based automated quantization for ReRAM-based hardware accelerator, which uses discrete action space to avoid the action space conversion step. We define a new reward function to enable the PPO agent to automatically learn the optimal quantization policy that meets the resource constraints, and give software-hardware modifications to support mixed-precision computing. Experimental results show that compared with coarse-grained quantization, the proposed method can reduce hardware cost by 20%~30% with negligible loss of accuracy. Compared with other automatic quantification, the proposed method has a shorter search time and can further reduce the hardware cost by about 4.2% under the same resource constraints. This provides insights for co-design of quantization algorithm and hardware accelerator.

Key words automated quantization; reinforcement learning; ReRAM-based hardware accelerator; neural network; processing in memory

摘 要 卷积神经网络在诸多领域已经取得超出人类的成绩.但是,随着模型存储开销和计算复杂性的不断增加,限制处理单元和内存单元之间数据交换的"内存墙"问题阻碍了其在诸如边缘计算和物联网

收稿日期:2021-06-07;**修回日期**:2021-09-08
基金项目:国家重点研发计划项目(2016YFB0200902)
This work was supported by the National Key Research and Development Program of China (2016YFB0200902).
通信作者:张兴军(xjzhang@xjtu.edu.cn)

等资源受限环境中的部署.基于阻变存储的硬件加速器由于具有高集成度和低功耗等优势,被广泛应用于加速矩阵-向量乘运算,但是其不适合进行32 b浮点数计算,因此需要量化来降低数据精度.手工为每一层确定量化位宽非常耗时,近期的研究针对现场可编程门阵列(field programmable gate array, FPGA)平台使用基于深度确定性策略梯度(deep deterministic policy gradient, DDPG)的强化学习来进行自动量化,但需要将连续动作转换为离散动作,并通过逐层递减量化位宽来满足资源约束条件.基于此,提出基于近端策略优化(proximal policy optimization, PPO)算法的阻变存储硬件加速器自动量化,使用离散动作空间来避免动作空间转换步骤,设计新的奖励函数使PPO自动学习满足资源约束的最优量化策略,并给出软硬件设计改动以支持混合精度计算.实验结果表明:与粗粒度的量化相比,提出的方法可以减少20%～30%的硬件开销,而不引起模型准确度的过多损失.与其他自动量化相比,提出的方法搜索时间短,并且在相同的资源约束条件下可以进一步减少约4.2%的硬件开销.这为量化算法和硬件加速器的协同设计提供了参考.

关键词　自动量化;强化学习;基于阻变存储的硬件加速器;神经网络;内存计算

中图法分类号　TP183

卷积神经网络(convolutional neural network, CNN)在图像分类、语音识别和自然语言处理等领域取得广泛应用[1-3].但是,如图1所示,模型准确度的提升是以设计更深、更复杂的模型为代价,这导致了更大的模型文件和更高的计算复杂度[4].在使用基于冯•诺依曼架构的通用处理器进行训练和推理时,"内存墙"问题(memory wall)限制了CPU和片外存储之间的数据传输,成为性能瓶颈.此外,文献[5]的研究表明CPU和片外存储进行一次数据传输的能耗比进行一次32 b浮点数加法运算高约2个数量级,大量的数据传输将产生巨大的能耗,这在边缘计算和物联网等资源受限的应用场景下是不可行的,因此,需要新的体系结构来缓解"内存墙"问题.

内存计算(processing in memory, PIM)通过将计算单元和存储单元紧密结合,从而缓解了"内存墙"的问题[6].基于忆阻器的交叉阵列(memristor-based crossbar),由于具有高密度、低功耗等优势,被广泛应用于加速矩阵-向量乘运算(matrix-vector multiplication, MVM).近期,专注于硬件设计的工程师提出了许多基于非易失性阻变存储(resistive RAM, ReRAM)的硬件加速器设计[7-9].但是,由于数模(digital-to-analog converter, DAC)和模数(analog-to-digital converter, ADC)转换器件的分辨率(resolution)、ReRAM cell的精度和ReRAM交叉阵列的尺寸有限,基于ReRAM的加速器无法直接高效地进行32 b浮点数运算[10].因此,需要使用低精度数值(low-precision data).

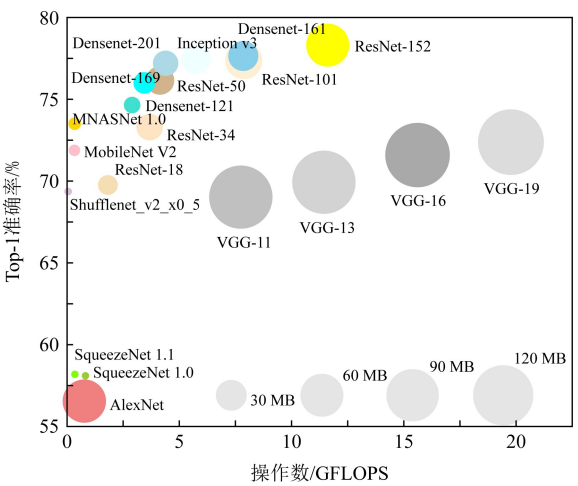


Fig. 1　Top-1 accuracy results of various CNNs on ImageNet

图1　CNN模型在ImageNet上的Top-1准确率

模型量化通过降低数据位宽,从而减少计算存储开销,这对基于ReRAM的硬件加速器非常友好[10].ISAAC[7]和PipeLayer[9]都使用16 b的权值和激励,Prime[8]使用8 b权值和6 b激励.但是,这些专注于硬件设计的工作都为权值和激励分配统一的量化位宽,这种粗粒度的模型级量化忽略了神经网络不同层的结构和冗余,无法有效达到模型精度和硬件开销的最佳性能折中.因此,需要更细粒度的量化策略,例如逐层量化(layer-wise quanti-zation).逐层量化的问题使其量化策略空间巨大.假设一个模型有L层(L指卷积层和全连接层的总和),每层的权值和激励可采用M种量化位宽的选择,那么其量化策略的搜索空间为$O(M^{2L})$.当M和L很大时,在如此大的量化策略空间上手工为每层搜索量化位宽非常耗时.近期研究提出使用自动机器学习(automated machine learning, AutoML)来自动选择

量化位宽.文献[11-13]使用基于深度确定性策略梯度(deep deterministic policy gradient, DDPG)[14]的强化学习算法来进行自动量化.但是这些研究存在3个问题:

1) 量化动作本质上是离散数值,而DDPG算法适用于连续动作,因此基于DDPG的方法[11-13]需要动作空间转换步骤;

2) 在文献[11]中,通过手工递减位宽来满足资源约束条件,而不是通过学习获得,并且其搜索时间漫长;

3) 大部分基于强化学习的自动量化是基于FPGA[11-12]和传统冯·诺依曼架构的通用处理器的[15],缺少结合自动量化算法的ReRAM加速器软硬件设计说明.

此外,由于没有成熟的ReRAM加速器芯片,目前主要基于模拟器来评估ReRAM加速器的性能.文献[16-17]提供详细的电路级(circuit-level)仿真,可用于仿真ReRAM加速器的面积、时延和功耗.文献[18]提出一个快速评估ReRAM硬件性能的行为级(behavior-level)模拟器,支持混合精度计算,但不支持自动选择量化位宽.

基于上述分析,本文工作的主要贡献有4个方面:

1) 提出基于近端策略优化(proximal policy optimization, PPO)[19]的自动量化算法,使用离散动作空间,避免动作空间转换步骤;

2) 设计新的包含ReRAM加速器硬件开销和模型精度的奖励函数,使PPO Agent通过学习来自动搜索同时满足资源约束和精度需求的量化策略,不需要手工递减量化位宽步骤;

3) 在遵循ReRAM加速器设计原则的前提下,结合自动量化算法,分析支持混合精度计算的ReRAM加速器的软硬件设计改动;

4) 实验表明:与粗粒度的量化方法相比,提出的方法可以减少20%~30%的硬件开销,而不引起模型准确度的过多损失.与其他自动量化相比,提出的方法自动搜索满足资源约束条件的量化策略,避免手工递减步骤,搜索时间快,并且在相同的资源约束条件下可以进一步减少约4.2%的硬件开销.

1 背景与相关工作

1.1 CNN推理阶段workload分析

为了不产生歧义,首先统一CNN和神经网络(neural network, NN)的术语:

1) CNN卷积核(filter or kernel)的权值(weight)对应于NN中的突触(synapse);

2) CNN中输入/输出特征图(ifmap/ofmap)的激励(activation)对应于NN中的输入/输出神经元(input/output neuron).

CNN推理阶段涉及卷积、池化、非线性激活(sigmoid or ReLU)和归一化等运算.推理阶段的计算负载(workload)主要集中在卷积层和全连接层.式(1)给出了卷积的计算公式,表1列出了相应的参数说明:

Table 1 Parameters of A CONV/FC Layer

表1 CONV/FC层参数说明

参数	说明
N	批大小
C_{out}	卷积核数或输出特征图的通道数
C_{in}	特征图数或卷积核的通道数
$H_{\text{in}}/W_{\text{in}}$	输入特征图尺寸信息
$K_{\text{height}}/K_{\text{width}}$	卷积核的尺寸信息
U	滑动步长

值得注意的是:

$$\begin{aligned}O[u][x][y]=&\\ B[u]+\sum_{k=0}^{C_{\text{in}}}\sum_{i=0}^{K_{\text{width}}}\sum_{j=0}^{K_{\text{height}}}&I[k][Ux+i][Uy+j]\times\\ &W[u][k][i][j],\\ 0\leqslant u&\leqslant C_{\text{out}},\\ 0\leqslant x&\leqslant (W_{\text{in}}-K_{\text{width}}+U)/U,\\ 0\leqslant y&\leqslant (H_{\text{in}}-K_{\text{height}}+U)/U.\end{aligned}\tag{1}$$

全连接层可以看成是当式(1)中参数$H_{\text{in}}=K_{\text{height}}$,$W_{\text{in}}=K_{\text{width}}$,$U=1$时的卷积层.因此,下面主要分析卷积计算的特点.结合图2绘制的卷积运算的示意图和式(1)分析可得,卷积层计算表现出2个特征:

1) 不连续的内存访问.这是由于卷积核在输入特征图上滑动时,滑窗(sliding window)内的数据是输入特征图的多行数据.因此,不论数据是按行存储还是按列存储,在不进行数据布局优化的条件下,滑窗内的数据内存访问都是不连续的.

2) 数据复用性.卷积运算中的数据复用主要分为权值复用(weight reuse)和输入复用(input reuse).权值复用表现为同一个卷积核的权重被多次用于输入特征图的不同区域.输入复用表现为输入特征图的同一个区域的激励被多次用于不同的卷积核.

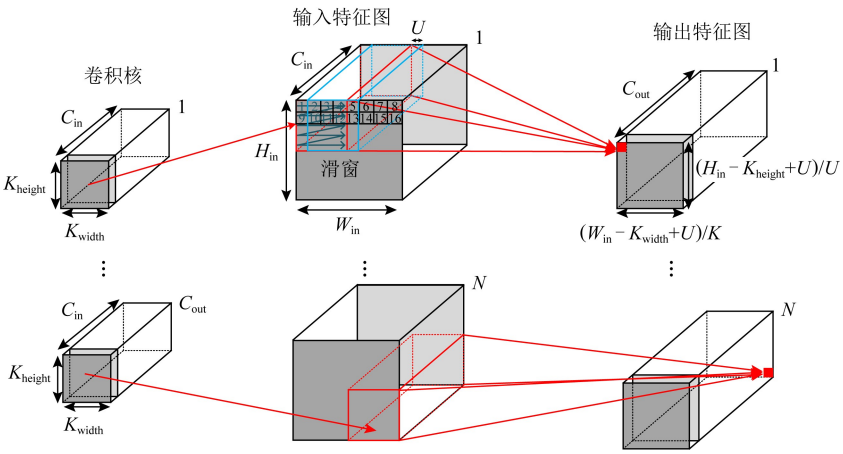


Fig. 2　Convolutions in CNNs

图 2　CNN 中的卷积运算

直接使用 for 循环实现的卷积计算会引起非常高的缓存缺失,从而导致运算速度非常慢.因此,通常将卷积计算转化为矩阵-矩阵乘(matrix-matrix multiplication)或矩阵-向量乘(matrix-vector multiplication, MVM).现有的深度学习框架(如 Tensorflow 和 Pytorch)底层都是通过调用通用矩阵乘库(general matrix multiply, GEMM)来实现高效卷积计算.此外,为减少不连续的内存访问并提高数据复用,在调用之前 GEMM 前,需要对数据进行转换和重排,最常见的转换方法是如图 3 所示的 im2col,这是一种用空间换时间的优化策略.

除了在算法层面上优化卷积运算,实现高效矩阵-向量乘也成为硬件加速器设计的关键.TPU[20]作为工业界最成功的专用硬件加速器,采用 65 536 个 8 b 乘法器和加法器,组成脉动阵列(systolic array)架构的矩阵乘法单元,每秒峰值速度为 92TFLOPS.TPU 在进行计算时,首先将权值参数从内存加载到乘法器和加法器阵列中.然后,从内存加载输入数据,每次执行乘加运算时,都会将结果传递给后面的乘法器,同时进行求和.与传统的基于 CMOS 实现的加速器相比[21-23],基于 ReRAM 的硬件加速器的本质是在模拟域实现矩阵-向量乘运算,因此具有速度快、功耗低等优势.下面介绍基于 ReRAM 的硬件加速器的基本原理.

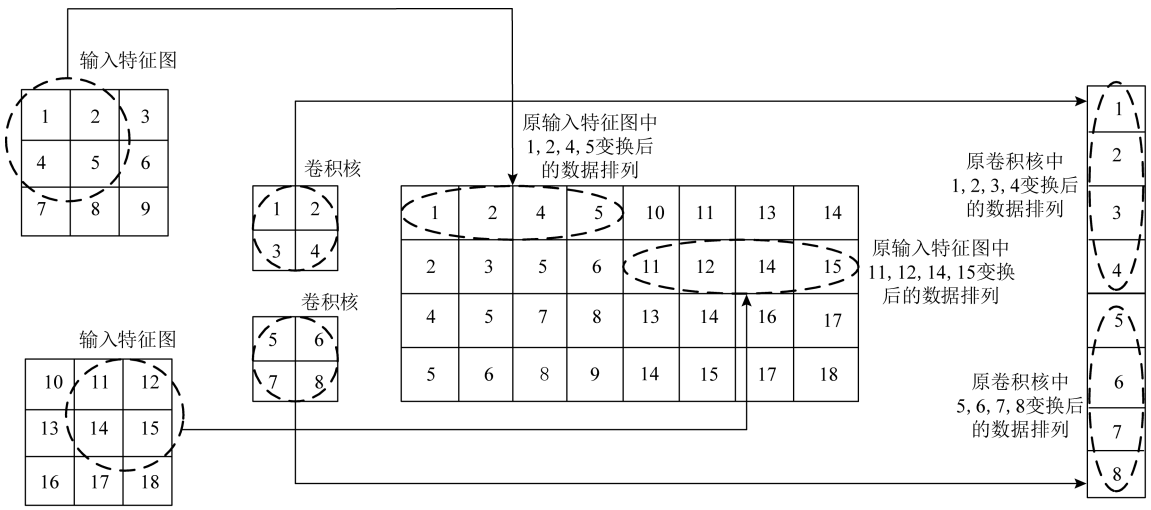


Fig. 3　The data rearrange of im2col

图 3　im2col 的数据重排原理图

1.2　基于 ReRAM 的硬件加速器设计

基于 ReRAM 的硬件加速器通常采用分层结构(hierarchical architecture)[7-9].如图 4 所示,加速器由多个 Tile 和 I/O 组成,I/O 用于从片外存储加载

数据,并通过基于片上通信网络(network on chip, NoC)进行 Tile 间的数据交换.数据的编排和通信由控制器(controller)负责.Tile 由缓冲(buffer)、移位-加法器(shift & adder)、用于计算非线性操作的特殊功能单元(special function unit, SFU)和多个 PE(processing engine)组成.PE 由输入/输出寄存器、移位-加法器、模数转换器(ADC)和多个 ReRAM 交叉阵列(ReRAM crossbar array, XB)组成.在推理阶段,输入数据通过 DAC 转换为模拟域的电压信号,便可通过基尔霍夫定律(Kirchoff's law),在一个读操作的延迟内完成一次矩阵-向量乘法计算,之后通过 ADC 将模拟电流信号转为数字信号.和推理相比,神经网络的训练涉及误差和梯度的计算与更新,需要设计额外的外部电路和 ReRAM 交叉阵列,这使得 ReRAM 硬件加速器的电路级设计更加复杂.此外,训练阶段频繁地更新操作造成大量的写操作,这给 ReRAM 的写耐久性带来巨大挑战[24-26].目前,大部分研究使用基于 ReRAM 的加速器来加速神经网络的推理.ISAAC[7]结合新的数据编码,提出了支持原位乘加操作的基础单元(in-situ multiply accumulate, IMA),并使用基于 ReRAM 的块内流水线(intra-tile pipeline)设计加速推理.Prime[8]分别设计用于计算(FF subarray)、存储(mem subarray)和缓存(buffer subarray)的 3 个阵列,避免使用 eDRAM 缓存和输入输出寄存器.和 Prime、ISAAC 不同,PipeLayer[9]通过合理的数据复用、层内并行(intra-layer parallelism)和层间流水线(inter-layer pipeline)来提高吞吐量.但是,这些硬件设计都使用较高的数据精度,并且为所有层指定统一的量化位宽,忽略了神经网络不同层的冗余.

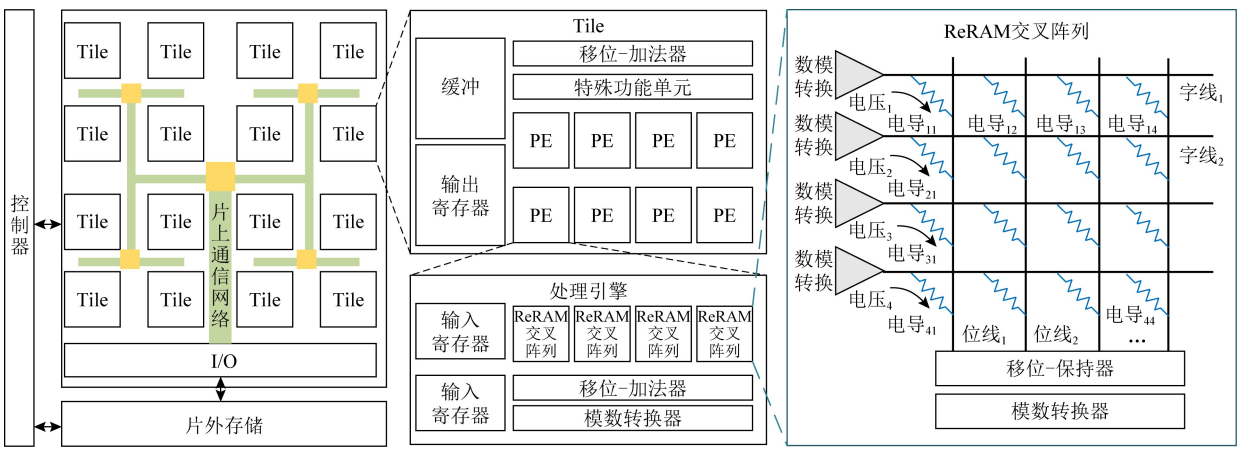


Fig. 4 The hierarchical architecture of ReRAM-based accelerator

图 4 基于 ReRAM 加速器的分层体系结构

为了快速评估 ReRAM 加速器的性能,研究人员提出了许多模拟器.文献[16-17]使用 C++实现 ReRAM 加速器的电路级(circuit-level)仿真软件,可用于仿真 ReRAM 加速器设计的面积、时延和功耗,但不支持混合精度计算.文献[18]使用 Python 实现一个快速评估 ReRAM 硬件性能的行为级(behavior-level)模拟器,支持混合精度计算,可用于芯片早期设计阶段的快速仿真,但不支持自动选择量化位宽.

1.3 量化与 AutoML

专注于 ReRAM 加速器设计的硬件工程师通常使用模型级量化.文献[16-17]都采用 W2A8 的量化策略,即将所有卷积层和全连接层的权值量化为 2 b 整型,激励量化为 8 b 整型.但是,这会导致模型准确度下降.手工为每一层确定量化位宽非常耗时.最近,专注于算法研究的工程师提出使用自动机器学习来自动选择量化位宽.文献[11]针对 FPGA 平台,提出了基于 DDPG 算法的硬件感知自动量化技术;文献[12]在文献[11]的基础上,也针对 FPGA 提出基于 DDPG 的卷积核级别(kernel-wise)的自动量化;文献[13]也在文献[11]的基础上,提出使用基于 DDPG 2 阶段强化学习,分别用于提高精度和 ReRAM 的存储利用率;文献[15]针对通用处理器 CPU,使用强化学习来进行自动量化.由于 DDPG 适用于连续动作空间,而量化位宽本质上是离散数值.因此,文献[11-13]都需要动作空间转换步骤,这一转化步骤可以通过使用适用于离散动作空间的算法来避免;此外,文献[11]中的资源受限约束是通过手工递减量化位宽实现的,而不是自动学习的,这可以通过设计新的包含模型精度和硬件开销的奖励函

数来实现.文献[27]使用基于神经架构搜索(neural architecture search, NAS)的方法,同时调整神经网络结构和量化策略来优化 ReRAM 加速器性能.本文不调整神经网络结构,主要探索量化策略对 ReRAM 加速器硬件开销的影响.

2　基于 PPO 的自动量化

本文使用基于 PPO Agent 的 actor-critic 模型来进行自动量化.下面详细说明状态空间、动作空间、奖励函数、量化方法和 Agent 的相关实现细节.

1) 状态空间(state space).如表 2 所示,状态是一个 8 维的特征向量,除了 $a^l_{\mathrm{w/a}}$ 外,其余的 7 个维度的特征向量覆盖了 1.1 节分析的卷积层和全连接层(FC layer)的所有属性.值得注意的是,如果第 l 层是全连接层,则 $l=1$, $k^l_{\mathrm{height}}=k^l_{\mathrm{width}}=1$, $U^l=0$.每个维度的向量在进行学习前都归一化到[0,1].

Table 2　Embeddings of the State Space

表 2　状态空间中的元素

参数	说明
l	层索引号,表示第 l 层
c^l_{out}	第 l 层的卷积核数或输出特征图的通道数
c^l_{in}	第 l 层的特征图数或卷积核的通道数
$\langle h^l_{\mathrm{in}}, w^l_{\mathrm{in}}\rangle$	第 l 层的输入特征图尺寸信息
$\langle k^l_{\mathrm{height}}, k^l_{\mathrm{width}}\rangle$	第 l 层的卷积核的尺寸信息
U^l	第 l 层的滑动步长
$I^l_{\mathrm{w/a}}$	0 表示权值 w,1 表示激励 a
$a^l_{\mathrm{w/a}}$	Agent 输出的动作,范围为[2,8]

2) 动作空间(action space).PPO 算法可用于连续动作空间和离散动作空间,本文使用离散动作空间,并且也将量化动作 $a^l_{\mathrm{w/a}}$ 限制在[2,8].本文通过定义新的奖励函数来使 Agent 最终可以学到满足资源约束的策略,不需要文献[11]中的手工递减步骤.下面将介绍奖励函数的设计.

3) 奖励函数(reward function).本文的奖励函数:

$$cost=\alpha\times latency+\beta\times energy+\gamma\times power,$$

$$R=\begin{cases}\lambda\times(acc_{\mathrm{quant}}-acc_{\mathrm{W8A8}}), \text{ 若}\dfrac{cost_{\mathrm{quant}}}{cost_{\mathrm{W8A8}}}\leqslant th \text{ 或}\\ \quad acc_{\mathrm{quant}}-acc_{\mathrm{W8A8}}\leqslant\tau,\\ -1, \text{ 其他},\end{cases}\tag{2}$$

其中,acc 表示模型准确度;$cost$ 表示硬件开销,为时延、能耗和功率的加权平均;α,β,γ 为加权系数,$\alpha+\beta+\gamma=1$,用户可以修改式(2)中的加权系数 α, β,γ 来控制时延、能耗和功率对硬件开销的贡献;下标 W8A8 表示权值和激励采用 8 b 量化,下标 quant 表示采用自动量化;τ 表示精度损失限制;th 用来描述资源约束条件,表示自动量化模型与采用 W8A8 量化模型的硬件开销比例,由于量化动作 $a^l_{\mathrm{w/a}}$ 被限制在[2,8],th 的取值范围为$[cost_{\mathrm{W2A2}}/cost_{\mathrm{W8A8}},1]$;$\lambda$ 是衰减因子,设置 $\lambda=0.1$.奖励函数的意义是,当搜索到的量化策略对应的量化模型的精度损失超过 τ 或硬件开销大于资源约束 th 时,奖励函数 $R=-1$.为统一精度损失过重或硬件开销过大时获得的奖励为−1,反推可得,τ 应设为 $-1/\lambda=-10$;此外,奖励函数鼓励 Agent 在满足资源约束的条件下搜索最大化模型精度的量化策略.式(2)中使用 W8A8 而不使用全精度(FP32)模型的原因有 2 个:1)从量化算法考虑,与全精度模型相比,W8A8 量化不会造成过大的精度损失,并且对硬件十分友好;2)从 ReRAM 硬件设计考虑,32 b 的浮点数不适用于 ReRAM 加速器.此外,由于本文将量化动作空间限制在[2,8],W8A8 量化策略就是量化策略空间的上限.

4) 量化方法.训练好的神经网络权值数据类型是 32 b 浮点数.由于 ReRAM cell 精度和 ReRAM 交叉阵列(ReRAM crossbar array, XB)尺寸有限,需要更多的 ReRAM cell 来表示更高位宽的数据.假设 ReRAM 设备的精度为 1 b/cell,一个 x 位的整形数据将占用 x 个 cells.本文将浮点数量转换为 x 位有符号整数(signed integer):

$$\boldsymbol{w}_{\mathrm{q}}=\Delta\times round\left(\frac{clip(\boldsymbol{w},c)}{\Delta}\right),$$

$$\Delta=\frac{c}{2^{x-1}-1},\tag{3}$$

其中,$\boldsymbol{w}_{\mathrm{q}}$ 表示量化后的权值,Δ 表示量化步长,函数 $clip$ 用来将权值 $\boldsymbol{w}$ 截断到$[-c,c]$的范围内,$round$ 表示向下取整.由于激励是非负的,在对激励进行量化时,函数 $clip$ 会将激励截断到[0,c].参数 c 的值在训练过程中更新:

$$c^*=\arg\min_c D_{\mathrm{KL}}(\boldsymbol{w}\parallel\boldsymbol{w}_{\mathrm{q}}),\tag{4}$$

其中,D_{KL} 表示量化前的 $\boldsymbol{w}$ 和量化后 $\boldsymbol{w}_{\mathrm{q}}$ 的分布的 KL 散度,其值越小表示 $\boldsymbol{w}$ 和 $\boldsymbol{w}_{\mathrm{q}}$ 的分布越接近.

5) Agent.传统的基于策略梯度的方法很难确定步长(或学习率),若步长太小,学习很慢;若步长太大,容易学到不好的策略,模型很难收敛.为了解决这个问题,PPO 算法在 TRPO[28] 的基础上,将约束作为目标函数的正则化项,并通过新旧策略的

比值来衡量新旧策略间的差异,限制新策略的更新幅度.

本文使用带有截断目标函数的 PPO 算法(PPO-Clip)来最大化期望奖励.其中 actor 和 critic 网络的前 2 层结构一样,都包含 2 个隐含层,每层包含 256 个神经元.由于 actor 网络用来产生动作,因此,actor 网络的最后一层的神经元个数为 8−2+1=7,输出表示[2,8]上的概率分布;critic 网络的最后一层的神经元个数为 1,输出状态价值.在每个时间步,Agent 为神经网络的特定层选择量化动作.一个回合(episode)指 Agent 遍历完所有层,由于神经网络的层数有限,因此回合长度是有限的.在探索过程中,使用 ADAM[29] 优化器,其中 $\beta_1=0.9$,$\beta_2=0.999$;actor 和 critic 网络学习率分别设为 3×10^{-4} 和 10^{-3}.通过最小化目标函数来更新 actor 网络的参数 θ:

$$J^{clip}(\theta)=\hat{E}_t[\min(r_t(\theta)\hat{A}_t,clip(r_t(\theta),1-\varepsilon,1+\varepsilon)\hat{A}_t)],$$
$$r_t(\theta)=\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)},\tag{5}$$

其中,$\hat{A}_t$ 是估计优势函数(estimated advantage function),π_θ 和 $\pi_{\theta_{\text{old}}}$ 分别表示新策略和旧策略,$r_t(\theta)$表示新旧策略间的差异,ε 表示截断比率,本文设 $\varepsilon=0.2$,截断函数 $clip$ 的作用是控制新旧策略间的差异不要过大.critic 网络的参数 φ 通过最小化损失函数来更新:

$$Loss(\varphi)=\hat{E}_t[(V_\varphi(s_t)-\hat{R}_t)^2].\tag{6}$$

3 软硬件设计改动

自动量化后的模型的每一层的数据精度不一致,这就需要 ReRAM 加速器支持混合精度计算(mixed-precision computing).文献[30]设计了一个支持混合精度计算的 ReRAM 加速器设计,但是不支持自动量化.本节遵循文献[17]给出的 ReRAM 加速器设计指导,结合文献[18,30-31],介绍支持混合精度计算的 ReRAM 加速器软硬件设计改动.

3.1 硬件设计改动

权值存储和矩阵分割.当使用 ReRAM 交叉阵列存储权值时,需要解决 2 个问题:1)当权值的位宽高于 ReRAM cell 能表示的范围时,如何表示高比特权值;2)当权值矩阵超过 ReRAM 阵列大小时,如何存储整个权值矩阵.

对于权值存储,文献[31]指出 1 b ReRAM 设备更加稳定和可靠,其设备级和电路级的非理想特性对准确度的影响不大.因此,本文使用 1 b ReRAM.此外,由于 ReRAM 设备的电导是正数,无法直接表征负权值,本文使用 1 组 ReRAM 交叉阵列来分别存储正负权值.由于量化后的权值的位宽被限制在[2,8],为支持最大 8 b 权值的存储,需要 8 组 ReRAM 交叉阵列.

对于矩阵分割,当权值矩阵大于 ReRAM 交叉阵列的尺寸时,需要按照 ReRAM 交叉阵列的尺寸分割权值矩阵,每个 ReRAM 交叉阵列存储一部分权值.在这种情况下,需要融合各 ReRAM 交叉阵列的中间计算结果来获得该层最终的输出.给出存储整个神经网络所需要的 ReRAM 交叉阵列数:

$$num_{\text{XB}}=\sum_{l=1}^{L}2a_{\text{w}}^{l}\times\left\lceil\frac{c_{\text{in}}^{l}k_{\text{height}}^{l}k_{\text{width}}^{l}}{Size_{\text{XB}}}\right\rceil\times\left\lceil\frac{c_{\text{out}}^{l}}{Size_{\text{XB}}}\right\rceil.\tag{7}$$

数据转换模块分析值和激励的位宽会影响数据转换模块的开销.量化后的激励需要通过 DACs 转换为输入电压,然后进行模拟域的 MVM 运算.假设 DAC 的分辨率为 Res_{DAC},输入数据转化的总逻辑时钟数:

$$cycles=\sum_{l=1}^{L}\frac{a_{\text{a}}^{l}}{Res_{\text{DAC}}}\times S_{\text{times}}^{l},\tag{8}$$

其中,S_{times}^{l} 表示第 l 层卷积时滑窗的次数.式(9)给出了使用 1 b ReRAM 时,ReRAM 交叉阵列的每一列输出的电流信号对应的数字信号的位宽为

$$Q_{\text{out}}=Res_{\text{DAC}}+\text{lb}(Size_{\text{XB}})+1,\tag{9}$$

理想情况下,ADC 的分辨率 Res_{ADC} 应该等于 Q_{out} 才能保证将模拟电流精确转换为数字信号.假设 $Res_{\text{DAC}}=1$ b,ReRAM 交叉阵列的尺寸通常为{128,256,512},那么理想的 Res_{ADC} 应该为{9 b,10 b,11 b}.但是,高分辨率 DAC/ADC 的功耗很大,这会弱化 ReRAM 加速器的低功耗优势.因此,不宜选择较高分辨率的 ADC,但是,Res_{ADC} 和 Q_{out} 间的差将引起硬件一级的量化误差.文献[16,30]表明 ADC 的分辨率大于等于 6 b 时,由 ADC 引起的精度损失可以忽略,因此,后续实验中默认使用 1 b DAC 和 8 b ADC,忽略 ADC 引起的硬件一级的量化误差.同时为方便和其他 ReRAM 加速器公平比较,使用 128×128 的阵列.

其他组件.除了 MVM 操作外,ReRAM 加速器还要支持 ReLU 和最大池化(max pooling)运算.采用类似文献[30-31]的方法,使用 look-up-table

(LUT)来实现 ReLU,使用寄存器保存序列中的最大值来实现最大池化.此外,为支持混合精度计算,控制器需要有寄存器来存储量化位宽.

3.2　软件设计改动

基于 3.1 节分析,本文提出如图 5 所示的软件框架,其工作流程为:①用户(算法工程师或硬件设计工程师)给出模型精度和硬件约束要求;②使用基于 PPO Agent 的 actor-critic 模型来进行逐层自动量化,如图 5 中第 l 层的权值和激励分别量化为 6 b 和 4 b(简写为 W6A4);③待整个模型的量化策略确定后,对模型进行逐层量化;④通过分析每一层的结构信息和量化位宽,将权值矩阵分割,并进行数据映射;⑤通过模拟器评估 ReRAM 加速器的硬件开销,更新状态和奖励函数;⑥搜索结束,返回最优策略;否则重复步②～⑤.

根据分析,本文基于 OpenAI 开源的强化学习框架 Spinning Up[32] 实现支持自动量化的前端训练框架.为了与前段训练框架对接,本文在文献[18]的基础上,使用 Python 实现一个用于评估 ReRAM 加速器硬件开销的行为级模拟器.本文目前实现的 ReRAM 模拟器仅用于评估推理阶段的硬件开销,所使用的时延、能耗和功率模型与文献[18]类似.文献[11]使用查表的方式获取 FPGA 的硬件开销,而本文的框架中需要调用模拟器来评估 ReRAM 加速器硬件开销,因此,调用模拟器的次数将影响搜索时间(将在 4.2.2 节分析).

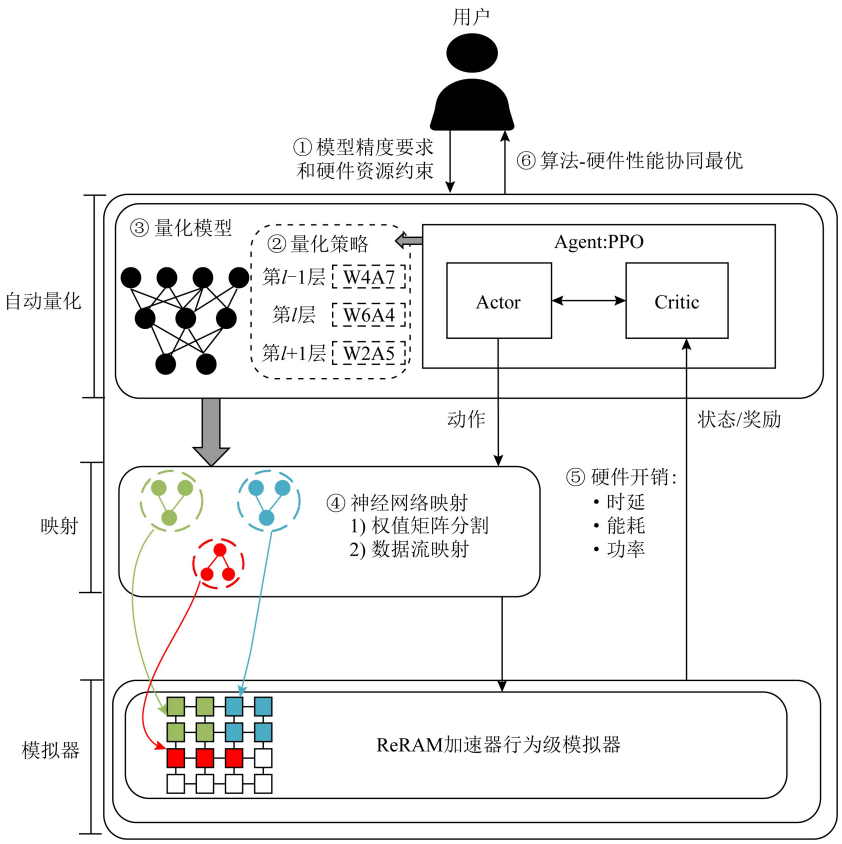


Fig. 5　The software framework of automated quantization

图 5　自动量化软件框架

4　实验结果与分析

4.1　实验环境与参数配置

实验的测试环境.本文使用的服务器配置信息为英特尔 Xeon Silver 4110 CPU@2.10 GHz,内存为 32 GB,GPU 为 NVIDIA Tesla P100,操作系统为 Ubuntu18.04.实验用的软件为 pytorch-1.6-gpu 和基于文献[18]实现的 ReRAM 模拟器.

表 3 列出了实验所用的数据集、神经网络模型.本文选择使用 LeNet-5 和 VGG-13 这 2 个不同规模的模型,分别在 MNIST 和 CIFAR-10 数据集上实验.由于标准的 VGG 模型是针对 ImageNet 数据集设计的,因此,本文修改 VGG 模型以适用于 CIFAR-10

数据集.LeNet-5 模型结构和文献[33]保持一致.表 4 列出了模拟器的配置信息,该配置是由 3.1 节分析得来的.本文使用 NVSim[34]来评估 ReRAM 阵列的硬件开销,ADCs/DACs 和减法器的开销直接从文献[7,35]获取,数字化电路和缓冲设计使用和文献[18]一样的评估方法.通过对所有组件的开销求和获得 ReRAM 加速器的硬件开销.

Table 3 The Configuration of the Neural Network

表 3 神经网络配置

网络名称	数据集	网络结构
LeNet-5	MNIST	与文献[33]方法一致
VGG-13	CIFAR-10	2×Conv3×128—pool— 2×Conv3×256—pool— 2×Conv3×512—pool— 1×Conv3×1024—pool— 1024—10

Table 4 The Default Configuration of the ReRAM Simulator

表 4 ReRAM 模拟器默认配置

参数	配置
交叉阵列尺寸	128×128
ReRAM 精度/(b/cell)	1
模数转换器	8 个,分辨率为 8 b,采样率为 1.2 GSPS
数模转换器	8 组,每组 128 个,分辨率为 1 b
移位-保持器	8
移位-加法器	8
减法器	8
Tile 上的 PE 数	16
PE 上的 XB 数	16
输入精度/b	2～8
权值精度/b	2～8

本文的对比实验主要包含 2 个方面:

1) 自动量化算法对比.由于基于 NAS 的方法改变神经网络结构,因此,为了公平比较,本文关注基于强化学习的自动量化算法.采用和文献[11]类似的设置,即神经网络的第 1 层和最后 1 层固定量化为 8 b.对比文献[36]提出的模型级量化和文献[11]提出的自动量化在不同资源约束下的模型精度和硬件开销.此外,针对文献[11]对比搜索时间.

2) ReRAM 硬件加速器对比.选择使用 16 b 数据精度的 ISAAC[7]和 PipeLayer[9],以及文献[31]提出的支持混合精度计算的加速器作为基线,对比功率效率(power efficiency).需要指出的是,本文的重点并不是要设计全新的 ReRAM 加速器,与使用相对较高数据精度的 ISAAC 和 PipeLayer 相比,本文强调使用自动量化来减少 ReRAM 硬件开销.

4.2 实验结果与分析

4.2.1 整体性能

为了证明所提方法的有效性,表 5 给出了在不同资源约束 *th* 下,所提出的方法与模型级量化[36]和基于 DDPG 的自动量化方法[11]的整体性能对比.从表 5 中可以看出,与文献[36]相比,所提出的自动量化方法可以减少 20%～30%的硬件开销.在相同的资源约束 *th* 下,所提出的方法的模型精度与文献[11]方法的相差不大,但是硬件开销 *cost* 小于文献[11]方法的.对于 LeNet-5 模型,所提出的方法主要减少 LeNet-5 模型在 ReRAM 加速器上的能耗和功率,但造成 1%～3%的精度损失.对于 VGG-13 模型,所提出的方法可以有效减少 VGG-13 模型在 ReRAM

Table 5 Overall Performance for Different NN Models Under Different Resource Constraints

表 5 不同资源约束下神经网络模型整体性能

th	方法	量化策略	指标	LeNet-5	VGG-13
1	文献[36]方法	W8A8	精度/%	98.11	92.66
			时延/ms	0.16	6.11
			能耗/mJ	0.002 3	6.67
			功率/W	0.80	139.78
0.7	文献[11]方法	自动	精度/%	95.55	92.49
			时延/ms	0.15	4.49
			能耗/mJ	0.001 6	3.38
			功率/W	0.34	114.39
			cost/%	68.61	68.67
	本文方法	自动	精度/%	95.13	92.27
			时延/ms	0.15	4.75
			能耗/mJ	0.001 6	2.99
			功率/W	0.32	116.01
			cost/%	**67.78**	**68.52**
0.8	文献[11]方法	自动	精度/%	96.66	92.60
			时延/ms	0.15	4.83
			能耗/mJ	0.001 7	4.30
			功率/W	0.55	126.65
			cost/%	78.80	78.00
	本文方法	自动	精度/%	96.63	**92.64**
			时延/ms	0.15	4.99
			能耗/mJ	0.001 7	3.54
			功率/W	0.53	116.91
			cost/%	**77.97**	**73.79**

注:黑体数据表示最好结果.

加速器上的硬件开销,且模型精度损失小于1%.与文献[11]方法相比,在相同的资源约束 th 下,所提出的方法比文献[11]方法可以多减少0.5%～4.21%的硬件开销.

图6显示了不同资源约束下的VGG-13模型精度在600个回合内的变化曲线,从图6中可以看出:1)当 $th=0.7$ 和 $th=0.8$ 时,模型精度在300个回合后收敛;但是当 $th=0.6$ 时,模型精度很难收敛,这是因为在资源约束相对严格时,PPO Agent很难学到同时满足精度要求和资源约束的策略.2) $th=0.8$ 时的模型精度整体上高于 $th=0.7$ 和 $th=0.6$ 时的模型精度,这是因为在资源约束相对宽松的条件,所设计的奖励函数鼓励PPO Agent提高量化位宽来提升模型精度.

图7显示了当 $th=0.6$,只考虑能耗($\beta=1,\alpha=\gamma=0$)时,VGG-13模型精度在600个回合内的变化曲线.从图7中可以看出,PPO Agent可以搜索到最优量化策略,并且模型精度收敛速度快(与图6中的虚线对比),这说明当硬件开销 *cost* 涵盖多个指标时(α,β,γ),优化变得更加困难.

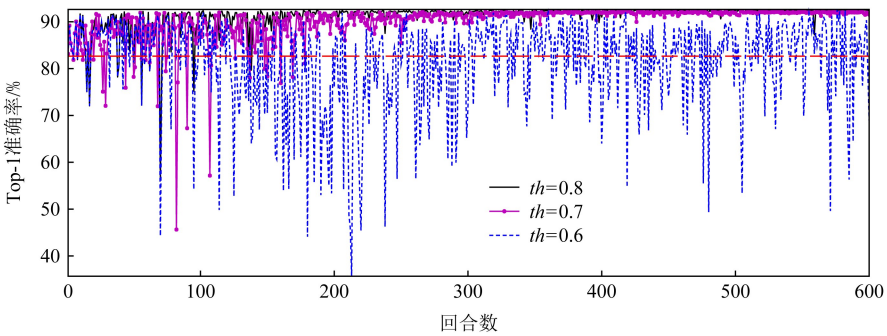


Fig. 6　Variation of Top-1 accuracy under different th

图6　不同资源约束下Top-1准确率的变化

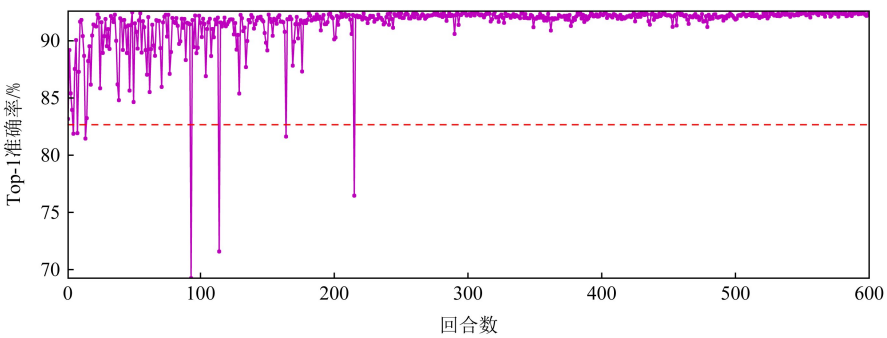


Fig. 7　Variation of Top-1 accuracy when $th=0.6,\alpha=\gamma=0,\beta=1$

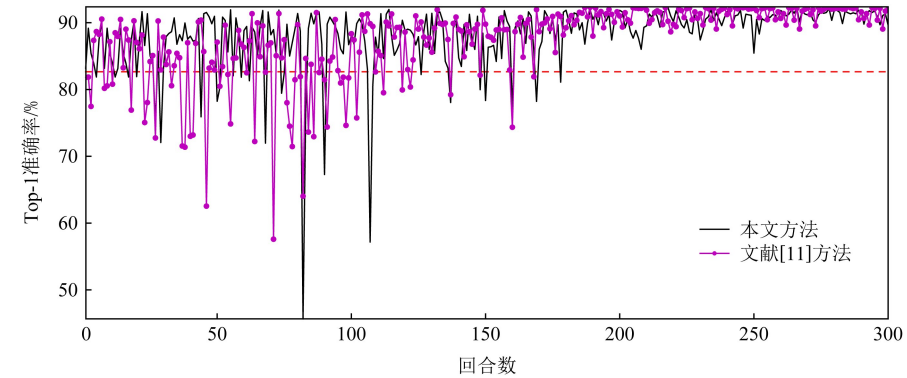
图7　当 $th=0.6$, $\alpha=\gamma=0$, $\beta=1$ 时Top-1准确率的变化

4.2.2　自动量化算法对比

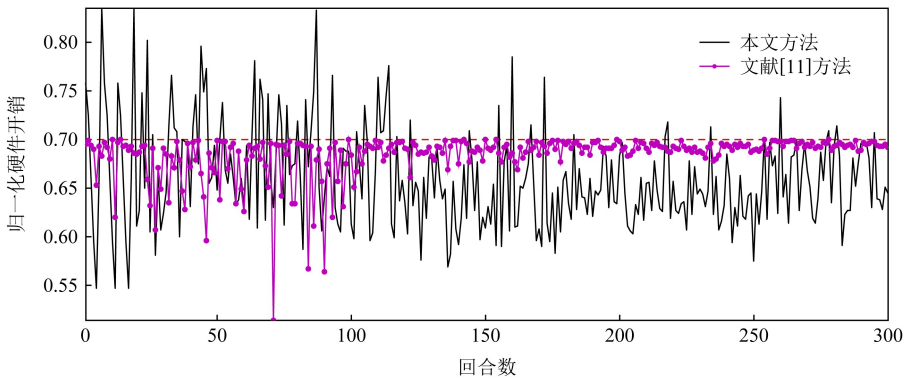
由于文献[12-13]都是在文献[11]的基础上进行的改进,因此本节主要和文献[11]进行对比.为了简化说明,本节以VGG-13模型在CIFAR-10数据集上的测试作为示例,从学习过程、量化策略和搜索时间3个方面进行对比.

1) 学习过程对比.图8显示了资源约束 $th=0.7$ 时,不同自动量化算法的模型精度和硬件资源开销在300个回合内的的变化曲线.从图8中可以看出,2种方法都能搜索到满足资源约束的量化策略,并保持模型精度.但是,图8(a)显示了本文方法在前200个回合搜到量化策略导致精度和硬件开销波动较大,探索能力更强,这是因为基于PPO Agent的动作是基于最近的随机策略采样获得的,而文献[11]是通过增加噪声来增强DDPG的探索;图8(b)清晰地显示了本文方法在200个回合后,可以学习到满足资源约束的量化策略,这得益于所设计的奖励函数,而文献[11]的奖励函数只与模型精度有关,需要手工递减量化位宽来满足资源约束.

2) 量化策略逐层对比.图9显示不同自动量化方法所搜索到的最优量化策略.从图9中可以看出,2种方法搜到的最优量化策略的不同主要表现在权



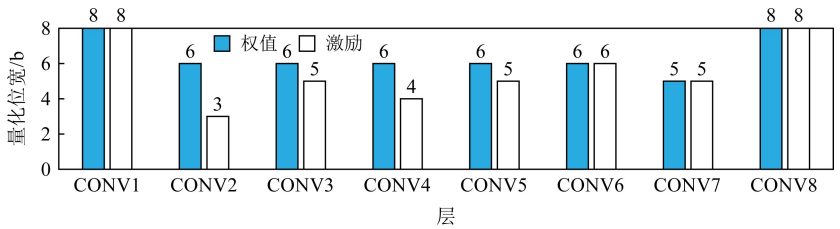
(a) 模型精度



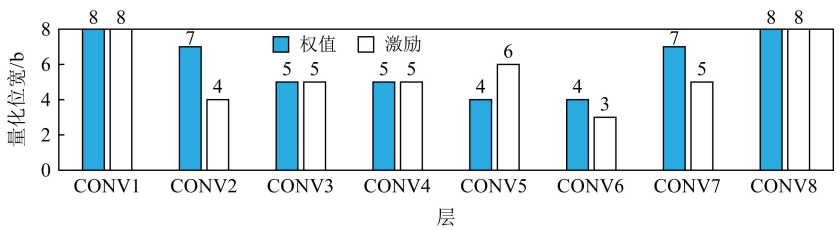
(b) 硬件开销

Fig. 8 Comparison of the learning procedure of different methods

图 8 不同方法的学习过程对比



(a) 文献[11]方法



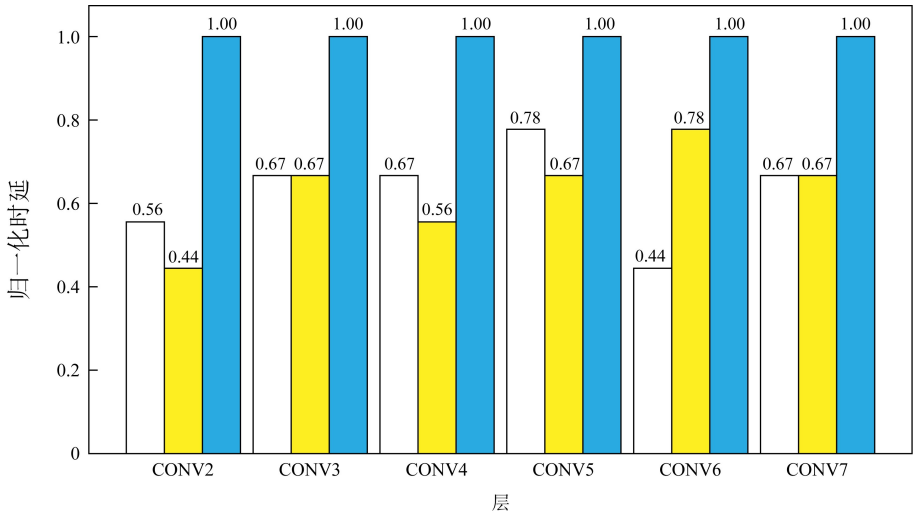
(b) 本文方法

Fig. 9 Comparison of the searched quantization policy of different methods under $th=0.7$

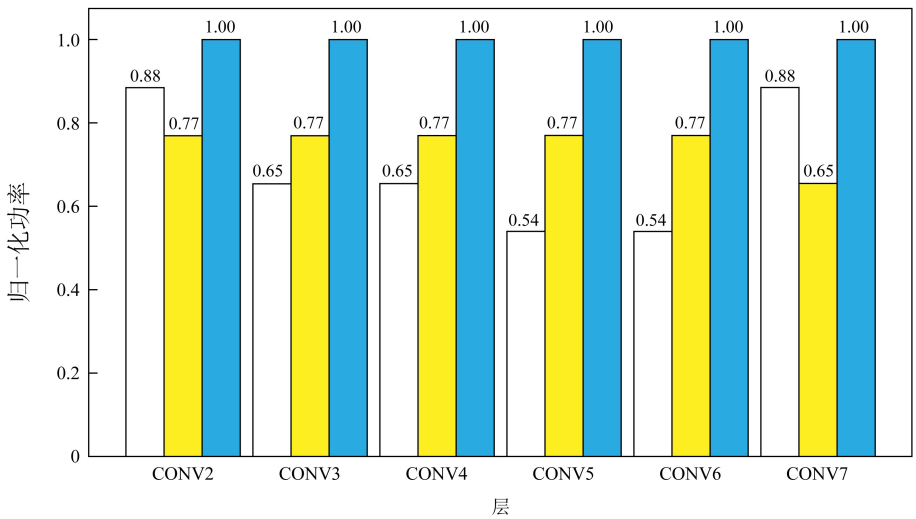
图 9 当 $th=0.7$ 时不同方法搜到的量化策略对比

值的量化位宽上.文献[11]为每一层的权值选择较高的量化位宽(大部分为 6 b),每一层的权值量化位宽平均为 6.38 b;而本文方法为每一层的权值选择的量化位宽的范围更大(4～7 b),每一层的权值量化位宽平均为 6 b,低于文献[11],因此当 $th=0.7$ 时,本文方法的硬件开销(68.52%)比文献[11]方法的(68.67%)少.

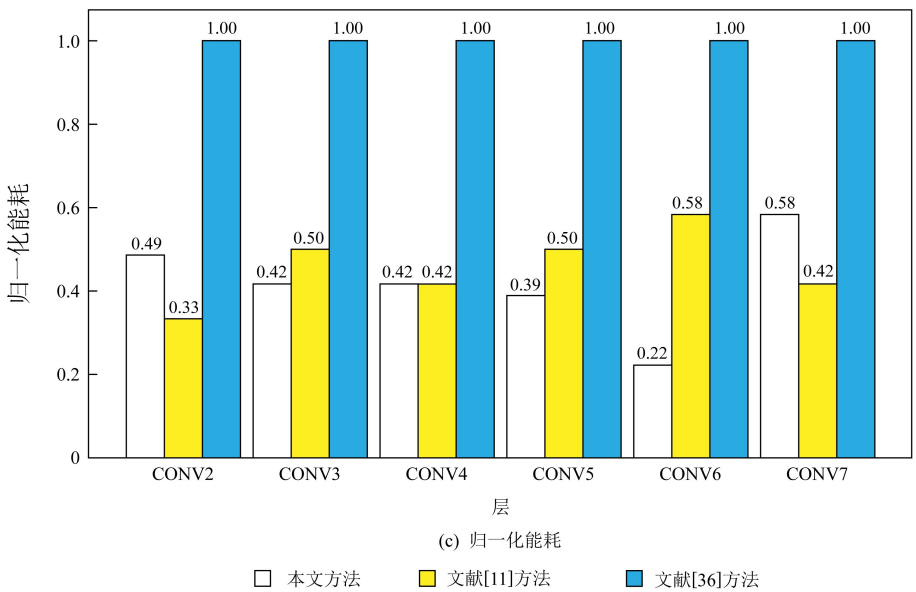
图 10 显示了图 9 中的量化策略对应的归一化



(a) 归一化时延



(b) 归一化功率



(c) 归一化能耗

Fig. 10　Layer-wise analysis of hardware cost of different methods

图 10　不同方法的硬件开销的逐层分析

的时延(latency)、功率(power)和能耗(energy),其值越小越好.从图 10(a)中可以看出,激励的量化位宽越高,时延越大.这与式(8)描述的一致.从图 10(b)中可以看出,权值的量化位宽越高,功率越大.从图 10(c)中可以看出,除了第 2 个卷积层(CONV2)和第 7 个卷积层(CONV7)外,本文方法比文献[11]方法能降低更多的能耗.

3) 搜索时间对比.表 6 显示不同自动量化方法的搜索时间对比.从表 6 中可以看出,本文所提出方法的搜索时间比文献[11]方法的少很多,这是由于本文方法在每个回合只调用一次模拟器来评估硬件开销,而文献[11]方法每递减一次量化位宽就需要调用一次模拟器来评估硬件开销.此外,文献[11]方法中的奖励函数鼓励 Agent 提高量化位宽,以此提高模型精度,但是这容易打破资源约束条件,导致需要更多的手工递减步骤.

Table 6 Comparison of Search Time

表 6 搜索时间对比

方法	th	回合	单次模拟用时/s	搜索时间/h
文献[11]方法	0.8	300	81	18
本文方法				8

图 11 展示了在 300 回合内,文献[11]方法在进行手工递减步骤前后的硬件开销的对比,其中有 131 个回合的量化策略超出了硬件约束 $th=0.8$,需要进行手工递减位宽.表 7 给出了第 43 回合的手工递减步骤前后的量化策略,结合图 11 和表 7 可看出,经过 7 次递减后,硬件开销 $cost$ 由 90.88%减到 79.17%,单在这一回合文献[11]方法比本文方法就需要多调用 7 次模拟器.当 th 更小时,这种情况更加严重.

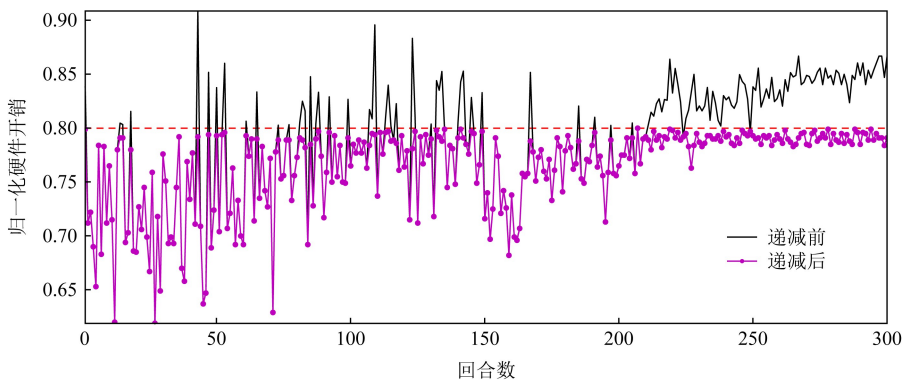


Fig. 11 Comparison of hardware cost before and after manual decrement step in ref[11]

图 11 文献[11]方法手工递减步骤前后硬件开销对比

Table 7 Comparison of Quantization Policy Before and After Manual Decrement Step in Epoch 43

表 7 第 43 回合的手工递减步骤前后量化策略对比

层	递减前		递减后		递减次数
	权值/b	激励/b	权值/b	激励/b	
CONV2	5	7	4	6	1
CONV3	8	7	7	6	1
CONV4	7	8	6	7	1
CONV5	7	6	6	5	1
CONV6	8	7	7	6	1
CONV7	8	7	6	5	2

4.2.3 和其他 ReRAM 加速器对比

表 8 对比了不同 ReRAM 加速器的功率效率.本文所提方法的功率效率为 454.8GOPS/(s×W),低于 ISAAC 的 627.5GOPS/(s×W),但高于 PipeLayer 的 142.9GOPS/(s×W).文献[31]提出支持混合精度计算的 ReRAM 加速器设计,并使用贪婪策略选择量化位宽.和文献[31]相比,本文提出的自动量化算法将 ReRAM 加速器的功率效率提升了近 1.50 倍.

Table 8 Comparison of Different ReRAM Accelerator Designs on Power Efficiency

表 8 不同 ReRAM 加速器设计功率效率对比

加速器设计	阵列尺寸	ADC 配置		功率效率/(GOPS/(s×W))
		分辨率/b	功率/mW	
ISAAC[7]	128×128	8	16	627.5
PipeLayer[9]	128×128	N/A	N/A	142.9
文献[31]方法	128×128	4	12	301.7
本文方法	128×128	8	16	454.8

注:PipeLayer 使用的是 Spiking,未使用 ADC. 数据取自文献[31]方法,本文方法的功率效率是基于 VGG-13 在 CIFAR-10 数据集上采用 $th=0.7$ 仿真获得的.

5 结　论

本文提出基于 PPO 的 ReRAM 神经网络加速器自动量化.使用 PPO Agent 来进行自动量化,通过设计新的奖励函数,实现了模型精度和硬件开销的最佳性能折中,并结合所提出的自动量化算法,给出 ReRAM 加速器的软硬件设计改动.实验结果表明:与模型级量化[36]相比,本文提出的方法可以减少 20%～30%的硬件开销.与文献[11]相比,本文提出的方法通过学习来自动搜索满足资源约束条件的量化策略,避免手工递减步骤,并且搜索时间快.与文献[31]相比,本文提出的方法将 ReRAM 加速器的功率效率提升了近 1.50 倍.这为量化算法和 ReRAM 加速器的协同设计提供了借鉴.

作者贡献声明:魏正提出研究思路,负责算法与实验设计,并撰写论文;张兴军负责技术方案设计与最终版本的修订;卓志敏负责行政和材料支持;纪泽宇负责方案讨论与论文校对;李泳昊负责分析数据与辅助实验.

参　考　文　献

[1] He Kaiming, Zhang Xiangyu, Ren Shaoqing, et al. Deep residual learning for image recognition [C] //Proc of the 29th IEEE Conf on Computer Vision and Pattern Recognition. Piscataway, NJ: IEEE, 2016: 770-778

[2] Lecun Y, Bengio Y, Hinton G. Deep learning [J]. Nature, 2015, 521(7553): 436-444

[3] Devlin J, Chang M W, Lee K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding [J].arXiv preprint, arXiv:1810.04805, 2018

[4] Deng Lei, Li Guoqi, Han Song, et al. Model compression and hardware acceleration for neural networks: A comprehensive survey [J]. Proceedings of the IEEE, 2020, 108(4): 485-532

[5] Han Song, Liu Xingyu, Mao Huizi, et al. EIE: Efficient inference engine on compressed deep neural network [J]. ACM SIGARCH Computer Architecture News, 2016, 44(3): 243-254

[6] Liu He, Ji Yu, Han Jianhui, et al. Training and software simulation for ReRAM-Based LSTM neural network acceleration [J]. Journal of Computer Research and Development, 2019, 56(6): 1182-1191 (in Chinese)
(刘鹤, 季宇, 韩建辉, 等. 面向阻变存储器的长短期记忆网络加速器的训练和软件仿真[J]. 计算机研究与发展, 2019, 56(6): 1182-1191)

[7] Shafiee A, Nag A, Muralimanohar N, et al. ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars [J]. ACM SIGARCH Computer Architecture News, 2016, 44(3): 14-26

[8] Chi Ping, Li Shuangchen, Xu Cong, et al. Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory [J]. ACM SIGARCH Computer Architecture News, 2016, 44(3): 27-39

[9] Song Linghao, Qian Xuehai, Li Hai, et al. Pipelayer: A pipelined reram-based accelerator for deep learning [C] //Proc of the 23rd IEEE Int Symp on High Performance Computer Architecture. Piscataway, NJ: IEEE, 2017: 541-552

[10] Sze V, Chen Yu-Hsin, Yang Tien-Hu, et al. Efficient processing of deep neural networks: A tutorial and survey [J]. Proceedings of the IEEE, 2017, 105(12): 2295-2329

[11] Wang Kuan, Liu Zhijian, Lin Yujun, et al. Haq: Hardware-aware automated quantization with mixed precision [C] // Proc of the 32nd IEEE Conf on Computer Vision and Pattern Recognition. Piscataway, NJ: IEEE, 2019: 8612-8620

[12] Lou Qian, Guo Feng, Liu Lantao, et al. Autoq: Automated kernel-wise neural network quantization [J]. arXiv preprint, arXiv:1902.05690, 2019

[13] Qu Songyun, Li Bing, Wang Ying, et al. RaQu: An automatic high-utilization CNN quantization and mapping framework for general-purpose RRAM accelerator [C/OL] // Proc of the 57th IEEE on Design Automation Conf(DAC). Piscataway, NJ: IEEE, 2020 [2021-08-06]. https://ieeexplore.ieee.org/document/9218724

[14] Lillicrap T P, Hunt J J, Pritzel A, et al. Continuous control with deep reinforcement learning [J]. arXiv preprint, arXiv: 1509.02971, 2015

[15] Elthakeb A T, Pilligundla P, Mireshghallah F S, et al. Releq: A reinforcement learning approach for deep quantization of neural networks [J]. arXiv preprint, arXiv: 1811.01704, 2018

[16] Chen Paiyu, Peng Xiaochen, Yu Shimeng. NeuroSim: A circuit-level macro model for benchmarking neuro-inspired architectures in online learning [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2018, 37(12): 3067-3080

[17] Zhang Wenqiang, Peng Xiaochen, Wu Huaqiang, et al. Design guidelines of RRAM based neural-processing-unit: A joint device-circuit-algorithm analysis [C/OL] //Proc of the 56th IEEE on Design Automation Conf(DAC). Piscataway, NJ: IEEE, 2019 [2021-08-06]. https://ieeexplore.ieee.org/document/8807058

[18] Zhu Zhenhua, Sun Hanbo, Qiu Kaizhong, et al. MNSIM 2.0: A behavior-level modeling tool for memristor-based neuromorphic computing systems [C] //Proc of the 30th Great Lakes Symp on VLSI. New York: ACM, 2020: 83-88

[19] Schulman J, Wolski F, Dhariwal P, et al. Proximal policy optimization algorithms [J]. arXiv preprint, arXiv: 1707.06347, 2017

[20] Jouppi N P, Young C, Patil N, et al. In-datacenter performance analysis of a tensor processing unit [C] //Proc of the 44th Annual Int Symp on Computer Architecture. New York: ACM, 2017: 1-12

[21] Moreau T, Chen Tianqi, Jiang Ziheng, et al. VTA: An open hardware-software stack for deep learning [J]. arXiv preprint, arXiv:1807.04188, 2018

[22] Chen Tianshi, Du Zidong, Sun Ninghui, et al. Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning [J]. ACM SIGARCH Computer Architecture News, 2014, 42(1): 269-284

[23] Chen Yu-Hsin, Krishna T, Emer J S, et al. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks [J]. IEEE Journal of Solid-State Circuits, 2016, 52(1): 127-138

[24] Zhang Wenqiang, Gao Bin, Tang Jianshi, et al. Neuro-inspired computing chips [J]. Nature Electronics, 2020, 3(7): 371-382

[25] Feinberg B, Wang Shibo, Ipek E. Making memristive neural network accelerators reliable [C] //Proc of the 24th IEEE Int Symp on High Performance Computer Architecture. Piscataway, NJ: IEEE, 2018: 52-65

[26] Ankit A, Hajj I E, Chalamalasetti S R, et al. PUMA: A programmable ultra-efficient memristor-based accelerator for machine learning inference [C] //Proc of the 24th Int Conf on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 2019: 715-731

[27] Jiang Weiwen, Lou Qiuwen, Yan, Zheyu, et al. Device-circuit-architecture co-exploration for computing-in-memory neural accelerators [J]. IEEE Transactions on Computers, 2020, 70(4): 595-605

[28] Schulmanchulman J, Levine S, Abbeel P, et al. Trust region policy optimization [C] //Proc of the 32nd Int Conf on Machine Learning. New York: ACM, 2015: 1889-1897

[29] Kingma D P, Ba J. Adam: A method for stochastic optimization [J]. arXiv preprint, arXiv:1412.6980, 2014

[30] Zhu Zhenhua, Sun Hanbo, Lin Yujun, et al. A configurable multi-precision CNN computing framework based on single bit RRAM [C/OL] //Proc of the 56th IEEE on Design Automation Conf (DAC). Piscataway, NJ: IEEE, 2019 [2021-08-06]. https://ieeexplore.ieee.org/document/8806777

[31] Cai Yi, Tang Tianqi, Xia Lixue, et al. Low bit-width convolutional neural network on RRAM [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2019, 39(7): 1414-1427

[32] Facebook OpenAI. Spinning Up [OL]. (2018-11-08) [2021-08-06]. https://github.com/openai/spinningup

[33] LeCun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition [J]. Proceedings of the IEEE, 1998, 86(11): 2278-2324

[34] Dong Xiangyu, Xu Cong, Xie Yuan, et al. Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2012, 31(7): 994-1007

[35] Han Song, Pool J, Tran J, et al. Learning both weights and connections for efficient neural networks [J].arXiv preprint, arXiv:1506.02626, 2015

[36] Choi J, Wang Zhuo, Venkataramani S, et al. Pact: Parameterized clipping activation for quantized neural networks [J]. arXiv preprint, arXiv:1805.06085, 2018

Wei Zheng, born in 1992. PhD candidate. His main research interests include computer architecture and deep learning.

魏 正,1992 年生.博士研究生.主要研究方向为计算机体系结构和深度学习.

Zhang Xingjun, born in 1969. PhD, professor, PhD supervisor. Member of CCF. His main research interests include high performance computing, big data storage system and machine learning acceleration.

张兴军,1969 年生.博士,教授,博士生导师.CCF 会员.主要研究方向为高性能计算、大数据存储系统和机器学习加速.

Zhuo Zhimin, born in 1982. PhD, professor. His main research interests include the overall design of defense systems and weapon systems.

卓志敏,1982 年生.博士,研究员.主要研究方向为防御体系与武器系统总体设计.

Ji Zeyu, born in 1986. PhD candidate. Student member of CCF. His main research interests include computer architecture, high performance computing and deep learning.

纪泽宇,1986 年生.博士研究生.CCF 学生会员.主要研究方向为计算机体系结构、高性能计算和深度学习.

Li Yonghao, born in 1993. Master candidate. His main research interests include high performance computing and deep learning.

李泳昊,1993 年生.硕士研究生.主要研究方向为高性能计算和深度学习.